**1.Write a blog on Difference between HTTP1.1 vs HTTP**

HTTP (Hypertext Transfer Protocol) is the foundation of the World Wide Web, allowing the transfer of data between a server and a client. HTTP1.1, released in 1999, was the standard for web communications for over a decade, until the release of HTTP2 in 2015. In this blog, we will discuss the differences between HTTP1.1 and HTTP2 in point by point.

1. Binary Format vs Text-based Format: HTTP1.1 is a text-based protocol, where data is transferred in ASCII characters. In contrast, HTTP2 uses a binary format, allowing for more efficient transfer of data.
2. Multiplexing: HTTP1.1 uses a single connection for each request/response cycle, which can cause latency and slow down page load times. HTTP2, on the other hand, allows for multiple requests to be sent and received over a single connection, resulting in faster page load times and better performance.
3. Header Compression: HTTP1.1 requires header information to be sent with each request, which can result in a significant amount of overhead. HTTP2 uses header compression to reduce the amount of data that needs to be sent, improving performance and reducing latency.
4. Server Push: HTTP2 allows for server push, where the server can send additional resources to the client before they are requested. This can reduce the number of roundtrips required and improve performance.
5. Flow Control: HTTP1.1 does not include flow control, which can lead to overloading and increased latency. HTTP2 includes flow control, allowing for better resource allocation and improved performance.
6. Security: HTTP1.1 is vulnerable to a range of security attacks, including man-in-the-middle attacks, data injection, and data tampering. HTTP2 includes built-in encryption, providing better security and protection against these types of attacks.
7. Compatibility: HTTP2 is designed to be backward-compatible with HTTP1.1, meaning that existing websites and applications should be able to support HTTP2 without any major modifications.

In conclusion, HTTP2 offers a range of improvements over HTTP1.1, including better performance, improved security, and backward compatibility. With its binary format, multiplexing, header compression, server push, flow control, and other features, HTTP2 is quickly becoming the standard for web communications.

**2. Write a blog about objects and its internal representation in Javascript?**

1. Introduction to Objects in JavaScript is an object-oriented language, meaning that it uses objects to store and manipulate data. Objects are essentially collections of properties that describe an entity, such as a person, a car, or a user interface element.
2. Creating Objects in JavaScript in JavaScript, objects can be created in several ways. The most common way is to use object literals, which are enclosed in curly braces {}. For example, here's an object literal that describes a person:

Code:

let person = { first name: "John", last name: "Doe", age: 30, hobbies: ["reading", "traveling", "hiking"] };

This object has four properties: first name, last name, age, and hobbies. The first name and last name properties are strings, age is a number, and hobbies is an array.

1. Internal Representation of Objects in JavaScript Internally, JavaScript represents objects as key-value pairs, where the key is a string and the value can be any data type, including another object. In fact, objects can have properties that are also objects, creating a nested structure.

When an object is created, JavaScript creates a new object in memory and assigns it a unique identifier. This identifier is used to reference the object in memory.

1. Accessing Object Properties Object properties can be accessed using dot notation or bracket notation. For example, to access the firstName property of the person object created earlier, you can use either of the following:

Code:

person.first name // returns "John" person["first name"] // also returns "John"

Dot notation is more commonly used, but bracket notation can be useful when the property name is stored in a variable.

1. Adding and Modifying Object Properties Object properties can be added or modified using either dot notation or bracket notation. For example, to add a new property to the person object, you can use either of the following:

Code:

person.gender = "male"; person["occupation"] = "developer";

1. Deleting Object Properties Object properties can be deleted using the delete operator. For example, to delete the age property of the person object, you can use the following:

Code:

delete person.age;

1. Object Methods Objects can also have methods, which are functions that are properties of an object. For example, here's an object that describes a car with a method that calculates its speed:

Code:

let car = { make: "Toyota", model: "Corolla", year: 2019, speed: 0, accelerate: function(amount) { this.speed += amount; } };

The accelerate method takes an amount parameter and adds it to the speed property of the car object using the this keyword, which refers to the current object.

In conclusion, objects are a fundamental concept in JavaScript that allows developers to store and manipulate data in a flexible and powerful way. By understanding the internal representation of objects and how to work with them, you can build more sophisticated and dynamic applications.